Efficient Formal Methods for the Synthesis of Concurrent Programs[2]

Paul C. Attie[3]
MIT Laboratory for Computer Science
Cambridge, MA
(attie@theory.lcs.mit.edu)

Introduction and background

The objective of this research is to produce useful, low-cost methods for developing correct concurrent programs from formal specifications. In particular, we address the design and verification of the synchronization and communication portions of such programs. Often, this portion can be implemented using a fixed, finite amount of synchronization related data, i.e., it is "finite-state." Nevertheless, even when each program component contains only one bit of synchronization related data, the number of possible global synchronization states for K components is about 2^K, in general. Because of this "state-explosion" phenomenon, the manual verification of large concurrent programs typically requires lengthy, and therefore error-prone, proofs. Using a theorem prover increases reliability, but requires extensive formal labor to axiomatize and solve verification problems. Automatic verification methods (such as reachability analysis and temporal logic model checking) use state-space exploration to decide if a program satisfies its specification, and are therefore also subject to state-explosion. To date, proposed techniques for ameliorating state-explosion either require significant manual labor, or work well only when the program is highly symmetric and regular (e.g., many functionally similar components connected in similar ways).

To overcome these drawbacks, we advocate the *synthesis of programs from specifications*. This approach performs the refinement from specifications to programs automatically. Thus, the amount of formal labor is reduced to writing a formal specification and applying the appropriate synthesis step at each stage of the derivation. While nontrivial, writing a formal specification is necessary in any methodology that guarantees correctness.

Our approach

Previous synthesis methods relied on some form of exhaustive search of the program's state-space. Hence, state-explosion has been a major obstacle to the application of synthesis to problems of realistic size. Our approach avoids exhaustive state-space search (and its exponential complexity) by analyzing interactions among every pair of component processes in the program separately, rather than looking at all processes at once. For every pair of directly interacting processes, we represent their interaction explicitly and separately from all the other interactions in the program. We start with a specification which describes all the pairwise interactions and we automatically synthesize a "pair-program" for each such interaction. We then "compose" the pair-programs together to produce the final large program. The final step refines this program (which uses shared memory) to a message passing model. To date, we have developed:

- A method for synthesizing fault-tolerant pair-programs [AAE98].
- An efficient method for composing pair-programs into a large (shared memory) program [AE98, Att99a]. This method allows all of the pair-programs to be different.
- A methodology for refining liveness properties [Att99b]. This forms the basis for refining the large shared memory program into a distributed program, which is a next step of our research.

Our approach has synthesized programs for resource contention problems such as K-process mutual exclusion and K-process dining philosophers, for arbitrary $K \geq 2$ [AE98], and also two-phase commit [Att99a].

Concluding remarks and future work

We aim to develop a methodology that can synthesize realistic concurrent programs. Our future work will increase the scope of applicability of our method, and will address non-functional properties such as fault-tolerance, performance, and distribution.

References

[AAE98] A. Arora, P. C. Attie, and E. A. Emerson. Synthesis of fault-tolerant concurrent programs. In *7th Annual ACM Symposium on the Principles of Distributed Computing*, pages 173 – 182, June 1998.

[AE98] P. C. Attie and E. A. Emerson. Synthesis of concurrent systems with many similar processes. *ACM TOPLAS*, 20(1):51–115, January 1998.

[Att99a] P. C. Attie. Synthesis of large concurrent programs via pairwise composition. In *CONCUR'99: 10th International Conference on Concurrency Theory*, number 1664 in LNCS. Springer-Verlag, August 1999.

[Att99b] P. C. Attie. Liveness-preserving simulation relations. In *18th Annual ACM Symposium on the Principles of Distributed Computing*, pages 63 – 72, May 1999.

[2]This work has been supported by NSF CAREER Grant CCR-9702616 and AFOSR Grant F49620-96-1-0221.

[3]On leave from the School of Computer Science, Florida International University.

Relational Programs[4]

Farokh B. Bastani
University of Texas at Dallas
Richardson, TX 77083
(bastani@utdallas.edu)

Motivation

Computers are being used to automate critical services, including manufacturing systems, transportation, etc. The software for these systems is becoming very complex due to their growing sophistication. For these critical applications, it is

[4]Part of this work has been supported by NSF grant CCR-9803993 and CCR-9900922.